CS18000: Problem Solving and Object-Oriented Programming
Selection
Video 1
The if statement
Selection

Booleans, Relations, and Selection Statements
Sequential Execution

• By default, a list of statements...
  – Statement 1;
  – Statement 2;
  – ...
  – Statement N;
• ...is executed in order, one after another
• ...unless there is an error ("exception"), all statements are executed
• We say, "Control flows sequentially."
Control Structures

- Language features (syntax) that affect the flow of control in a program
- Default control flow is sequential
- Control flow jumps to methods, then returns
- Various keywords introduce changes to the default flow
  
  if
  switch
  while
  for
The if Statement

if (boolean-expression)
    then-statement;
next-statement;

if (boolean-expression)
    then-statement;
else
    else-statement;
next-statement;
Decision Making

• If it is a weekday and I’m not on vacation, then I will get up early
• If there is a basketball game on and Purdue is playing, I’ll cheer for Purdue, otherwise if IU is playing, I’ll cheer for their opponent
Primitive Type: boolean (Review)

• Set of two elements \{ true, false \}

• Set of operations
  – Logical: && (and), || (or), ^ (xor), and ! (not)
  – Testing in various Java statements (e.g., if)

• Created by comparison operators
  – \( x < y, x \leq y, x = y, x \neq y, x > y, x \geq y \)
  – And result of logical operators (above)
  – Note: == and != also work with reference types, but only compare references (addresses) not the values
Comparing Strings

• `==` does not work in the way you might expect
• Strings are objects
• `==` between objects only compares the references (addresses) of the objects
• Two different String objects with the exact same characters will compare `==` false (since their objects are stored in different locations)
• if `(s1.equals(s2))`
  
  `then-statement;`
Comparing Strings

\[ S_1 \rightarrow \text{"Purdue"} \]

\[ S_2 \rightarrow \text{"Purdue"} \]
Abstracting Conditions

• If it is a weekday and I’m not on vacation, then I will get up early
  
  ```java
  if (isWeekday && !onVacation)
    getUpEarly();
  ```

• If there is a basketball game on and Purdue is playing, I’ll cheer for Purdue, otherwise if IU is playing, I’ll cheer for their opponent
  
  ```java
  if (gameOn(basketball) && playing(purdue))
    cheerFor(purdue);
  else if (gameOn(basketball) && playing(iu))
    cheerFor(opponent(iu));
  ```
Video 2
Examples using if statements
Problem: SecretWord

• Write a program that reads a word from the user and prints a message if it matches a “secret word” in the program.
import java.util.Scanner;

public class SecretWord {
    final static String SECRET = "awesome";

    public static void main(String[] args) {
        Scanner in = new Scanner(System.in);
        String word = in.next();
        if (word.equals(SECRET))
            System.out.printf("You have said the secret word: '%s'\n", SECRET);
    }
}
Problem: Absolute Value

• Write a program that illustrates how to convert the value in a variable x to the absolute value using an if statement
import java.util.Scanner;

public class AbsVal {
    public static void main(String[] args) {
        Scanner in = new Scanner(System.in);
        int x = in.nextInt();

        System.out.printf("BEFORE: x = %d\n", x);

        if (x < 0)
            x = -x;

        System.out.printf("AFTER: x = %d\n", x);
    }
}

Solution: Absolute Value
Solution: Absolute Value

\[ x = 37 \]
BEFORE: \( x = 37 \)
AFTER: \( x = 37 \)

\[ x = -41 \]
BEFORE: \( x = -41 \)
AFTER: \( x = 41 \)
Problem: DaisyDriveIn

• If you work more than 20 hours at the Daisy Drive-in, they pay you $8/hour for first 20 hours plus $10/hour for all hours above 20. Otherwise, they pay you $7/hour

• Write a method...
  
  double computePay(double hours)

• that returns the correct pay
public class DaisyDriveIn {
    double computePay(double hours) {
        if (hours > 20)
            return 8.00 * 20 + (hours - 20) * 10.00;
        else
            return hours * 7.00;
    }

    public static void main(String[] args) {
        DaisyDriveIn d = new DaisyDriveIn();
        double pay;

        pay = d.computePay(20); // pay should be 140
        pay = d.computePay(21); // pay should be 170
        pay = d.computePay(9.5); // pay should be 66.5
        pay = d.computePay(9.1); // pay should be 63.7
    }
}
Video 3
More complex if statements
More Selection Statements

A few more if-else examples
switch statement
## Boolean Operations

| A   | B   | A || B | A && B | A ^ B |
|-----|-----|-------|--------|-------|
| True| True| True  | True   | False |
| True| False| True  | False  | True  |
| False| True| True  | False  | True  |
| False| False| False | False  | False |
Problem: Median

• Write a method...
  double median(double a, double b, double c)

• This example illustrates cascading if-else-if statements
double median(double x, double y, double z) {
    if (x <= y && y <= z || z <= y && y <= x)
        return y;
    else if (y <= x && x <= z || z <= x && x <= y)
        return x;
    else if (x <= z && z <= y || y <= z && z <= x)
        return z;
    else
        return 0;
}
Solution: Median

\[ x = 12 \quad y = 14 \quad z = 27 \]
median is \( y \)

\[ x = 14 \quad y = 27 \quad z = 12 \]
median is \( x \)

\[ x = 24 \quad y = 15 \quad z = 18 \]
median is \( z \)
Basic Forms of the “if” Statement

if (boolean-expression)
    statement-if-true;

if (boolean-expression)
    statement-if-true;
else
    statement-if-false;
Blocks and Braces

• Use braces ({})) to group a sequence of statements into a single unit
• Already seen with method bodies and other examples
• Also can be use for control structures
Block Forms of the “if” Statement

```java
if (boolean-expression) {
    list-of-statements-if-true;
}

if (boolean-expression) {
    list-of-statements-if-true;
} else {
    list-of-statements-if-false;
}
```
Video 1
Examples using complex if statements
Problem: Swapper

• Write a program that, given two values in variables x and y, ensures that y is not less than x (swap them if necessary)
Solution: Swapper

```java
import java.util.Scanner;

public class Swapper {
    public static void main(String[] args) {
        Scanner in = new Scanner(System.in);

        int x = in.nextInt();
        int y = in.nextInt();

        System.out.printf("BEFORE: x = %d, y = %d\n", x, y);

        if (y < x) {
            int t = x;
            x = y;
            y = t;
        }

        System.out.printf("AFTER: x = %d, y = %d\n", x, y);
    }
}
```
Solution: Swapper

\[ x = 36 \quad y = 52 \]

**BEFORE:** \[ x = 36, \quad y = 52 \]

**AFTER:** \[ x = 36, \quad y = 52 \]

\[ x = 63 \quad y = 18 \]

**BEFORE:** \[ x = 63, \quad y = 18 \]

\[ t = 63 \quad x = 18 \quad y = 63 \]

**AFTER:** \[ x = 18, \quad y = 63 \]
Problem: Quadratic

• Write a method...
  ```java
  void printRoots(double a, double b, double c)
  ```
• that finds and prints the roots of a quadratic equation (including imaginary roots)
  ```latex
  ax^2 + bx + c = 0
  2x^2 - 9x + 4 = 0
  a=2 \quad b=-9 \quad c=4
  d = 81-32 = 49
  x1=4.0 \quad x2=0.5
  ```
public class Quadratic {
    void printRoots(double a, double b, double c) {
        double d = b * b - 4 * a * c;

        if (d < 0) {
            double x = -b/(2*a), xi = Math.sqrt(-d)/(2*a);
            System.out.printf("%.2f+%.2fi and %.2f-%.2fi are imaginary roots of %.2fx^2 + %.2fx + %.2f\n", x, xi, x, xi, a, b, c);
        } else {
            double x1 = (-b + Math.sqrt(d))/ (2 * a);
            double x2 = (-b - Math.sqrt(d))/ (2 * a);
            System.out.printf("%.2f and %.2f are real roots of %.2fx^2 + %.2fx + %.2f\n", x1, x2, a, b, c);
        }
    }

    public static void main(String[] args) {
        Quadratic q = new Quadratic();
        q.printRoots(3, 4, 5);
        q.printRoots(2, 4, -30);
        q.printRoots(12, 5, 3);
    }
}
Problem: FileExtensions

• Write a method...

String findExtension(String filename)

• that finds the extension in String filename.

• Example illustrates short-circuit evaluation
Problem: FileExtensions

String name = new String ("fluffy.jpg");

String ext = findExtension (name);

substring [begin, end)
[0.0, 100.0) starts at 0.0 and goes up to but does not include 100.0
[6,10) is 6, 7, 8, 9
String findExtension(String fname) {
    int dot;

    if (fname == null)
        return "";

    dot = fname.indexOf('.');

    if (dot == -1)
        return "";

    String extension = fname.substring(dot, fname.length());

    return extension;
}
Strings, Empty Strings, Null Pointers

String fname = new String("fluffy.jpg");
fname points to the object that contains the string "fluffy.jpg"

String fname = new String("");
fname points to the object that contains the string "" (empty string)

String fname = null;
fname does not point to a string. It is a null pointer.
String findExtension(String fname) {
    int dot;

    if (fname == null || fname.indexOf('.') == -1)
        return "";

    dot = fname.indexOf('.');

    String extension = fname.substring(dot, fname.length());

    return extension;
}
Short-Circuit Evaluation

• The Boolean operators && (and) and || (or) abandon evaluation if the result is determined with certainty, e.g., no matter what “whatever” is true || (whatever) -> must be true false && (whatever) -> must be false

• In these cases, “whatever” is not evaluated

• Common use:
  p != null && p.isImportant()
  – Second expression would cause null pointer exception if p were null
Video 2
Special cases with if statements
Dangling Else Problem

```java
if (familyVisiting)
    if (isWarmOut)
        goToPark();
else
    hangoutWithFriends();
```

- The else clause goes with the most recent if, not as formatted above
  ```java
  if (familyVisiting)
      if (isWarmOut)
          goToPark();
  else
      hangoutWithFriends();
  ```

- Use braces to disambiguate
Problem: ChangeBack

• Comparing real valued quantities for equality
• You gave the waiter a $10 bill
double paid = 10.00;
• The check was $9.10
double check = 9.10;
• The waiter gave you 90 cents back
double change = 0.90;
• Is it correct?
  (paid - check) == change
public class ChangeBack {
    double computeChange(double check, double paid) {
        return paid - check;
    }

    public static void main(String[] args) {
        ChangeBack c = new ChangeBack();
        double change;

        change = c.computeChange(8, 10); // 2.00
        change = c.computeChange(9.10, 10); // 0.90
    }
}
Comparing Real Values

• Since real numbers represented with double and float are imprecise...
• Comparisons may fail when they shouldn’t
• Common trick:
  – Replace...
    if (x == y)
  – By...
    if (Math.abs(x – y) < epsilon)
  – For some small value, epsilon
    if (Math.abs(c.computeChange(9.10, 10) – 0.90) < 0.001) {...}
Ternary Assignment Operator

• A common situation is to assign one of two alternative values depending on a condition

```java
if (a < b)
    minVal = a;
else
    minVal = b;
```

• We can use the following equivalent statement

```java
minVal = (a < b)? a : b;
```
Video 3
Switch statement
Problem: Days

• Write a method that returns the number of days in a given month from a given year

```java
int daysInMonth(int month, int year)
```
Solution 1: Days

```java
int daysInMonth1(int month, int year) {
    if (month == 1) // January
        return 31;
    else if (month == 2) {
        LeapYear ly = new LeapYear();
        if (ly.isLeapYear(year))
            return 29;
        else
            return 28;
    }
    else if (month == 3)
        return 31;
    else if (month == 4)
        return 30;
    else if (month == 5)
        return 31;
    else if (month == 6)
        return 30;
    else if (month == 7)
        return 31;
    else if (month == 8)
        return 31;
    else if (month == 9)
        return 30;
    else if (month == 10)
        return 31;
    else if (month == 11)
        return 30;
    else if (month == 12)
        return 31;
    return -1;
}
```
Solution 2: Days

```java
int daysInMonth2 (int month, int year) {
    switch (month) {
        case 1:  case 3:  case 5:  case 7:  case 8:  case 10:  case 12:
            return 31;
        case 4:  case 6:  case 9:  case 11:
            return 30;
        case 2:
            LeapYear ly = new LeapYear();
            if (ly.isLeapYear(year))
                return 29;
            else
                return 28;
    }
    return -1;
}
```
Solution 2: Days (arrow case labels)

```java
int daysInMonth2 (int month, int year) {
    switch (month) {
        case 1, 3, 5, 7, 8, 10, 12 -> {
            return 31;
        }
        case 4, 6, 9, 11 -> { // right hand side of the arrow has to be
            // an expression or block
            return 30;
        }
        case 2 -> {
            LeapYear ly = new LeapYear();
            if (ly.isLeapYear(year))
                return 29;
            else
                return 28;
        }
    }
    return -1;
}
```
Solution 2: Days (switch expressions, arrow case labels, yield statements)

```java
int daysInMonth2 (int month, int year) {
    int numberOfDays;
    numberOfDays = switch (month) {
        case 1, 3, 5, 7, 8, 10, 12 -> 31; // can just put value here
        case 4, 6, 9, 11 -> 30; // and here
        case 2 -> { // in a block must use "yield"
            LeapYear ly = new LeapYear();
            if (ly.isLeapYear(year))
                yield 29;
            else
                yield 28;
        }
        default -> -1:
    };
    // switch expression form has to end in a semicolon
    return numberOfDays;
}
```
Problem: what stuff

- Write a method that returns a string saying what stuff students are given based on their year in college:
  - Seniors (4) and Juniors (3) get a new backpack
  - Sophomores (2) get a new coat
  - Freshmen (1) get new gloves and a new coat

String whatStuff (int yearInCollege)
Solution: what stuff

String whatStuff (int yearInCollege) {
    String stuff = "You will be given ";
    switch (yearInCollege) {
        case 1:
            stuff = stuff + "new gloves ";
        case 2:
            stuff = stuff + "new coat";
            break;
        case 3: case 4:
            stuff = stuff + "new backpack";
            break;
        default:
            stuff = stuff + "nothing";
            break;
    }
    return stuff;
}
Solution: what stuff

stuff → "You will be given"

"You will be given" + "new backpack"

"You will be given new backpack"
Switch can use Strings

String whatStuff (String yearInCollege) {
    String stuff = "You will be given ";
    switch (yearInCollege) {
        case "Freshman":
            stuff = stuff + "new gloves ";
            break;
        case "Sophomore":
            stuff = stuff + "new coat";
            break;
        case "Junior": case "Senior":
            stuff = stuff + "new backpack";
            break;
        default:
            stuff = stuff + "nothing";
            break;
    }
    return stuff;
}